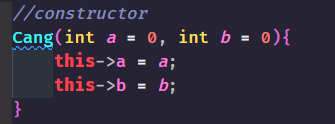
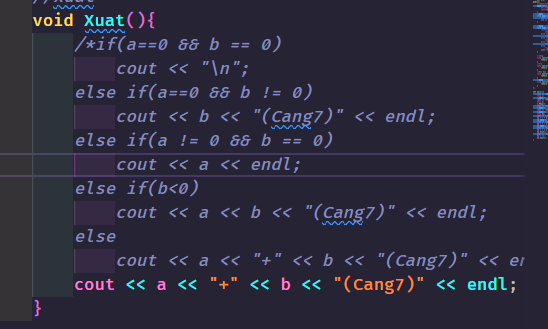
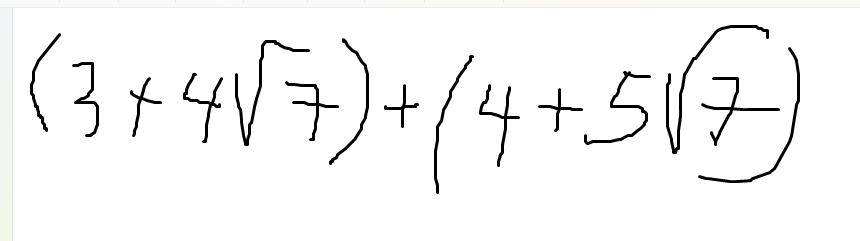
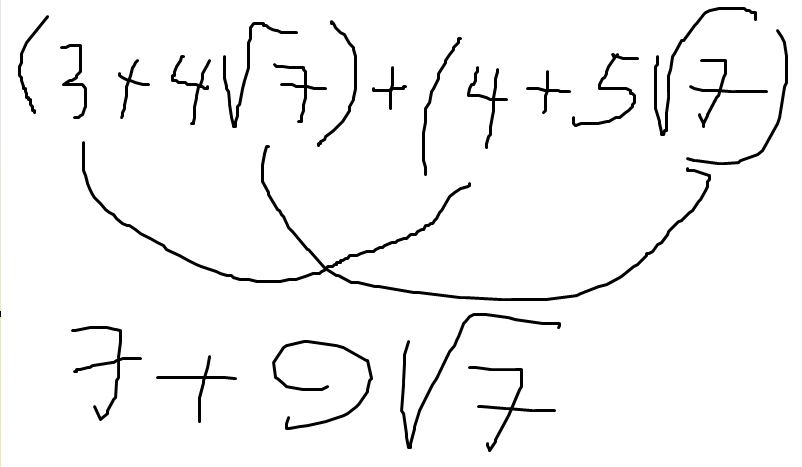
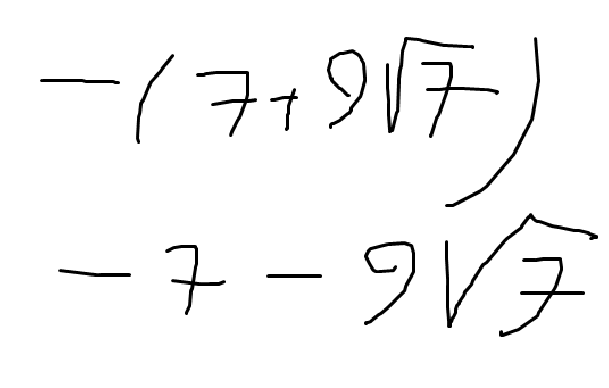
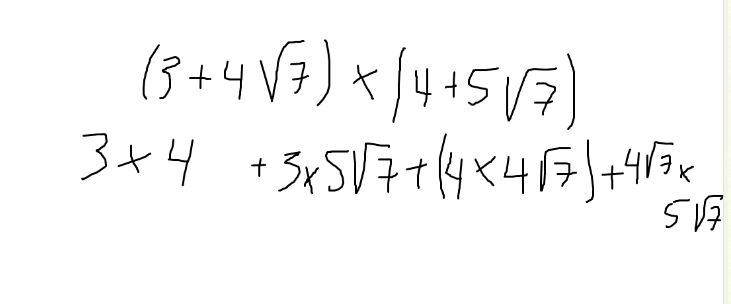
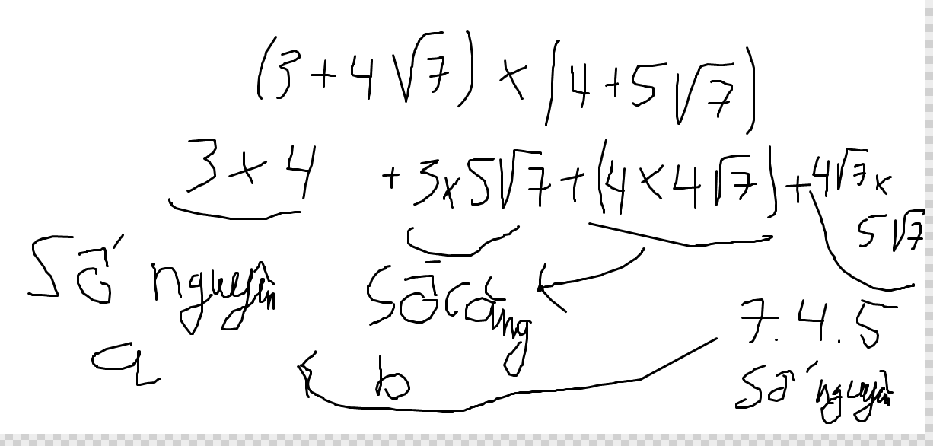
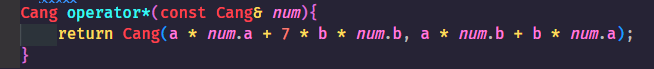
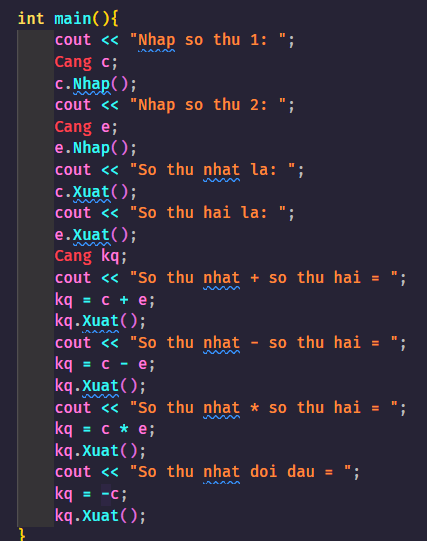
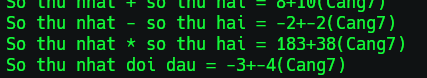
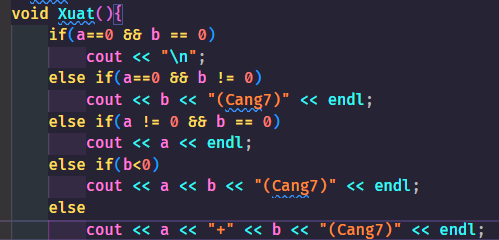
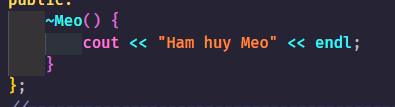
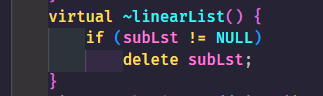
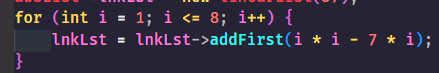
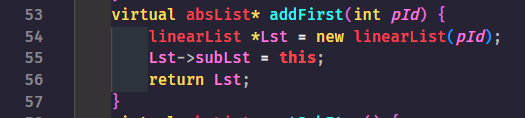
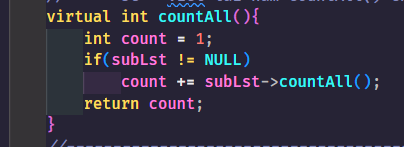
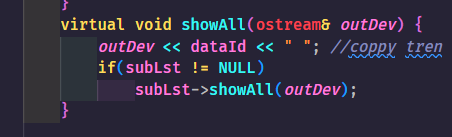
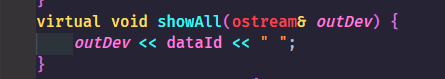
Câu 1: file 2013C1.cpp  
Hai câu a vs b gộp chung   
  
Class Cang   
  
đầu tiên constructor , mình gán mặc đình a = 0, b = 0 vì sẽ có nhứng trường hợp là không có a hoặc không có b,  sẽ là dạng giống như vậy nếu mình khai báo  
  
Hàm xuất : nếu mà nói cơ bản thì chỉ cần cout ra tí nữa mình sẽ phân tích tại sao note lại phần này  
  
Phép cộng:   
operator+ : const num để tí nữa nếu return về giá trị Cang thì sẽ không bị thay đổi, không ảnh hưởng tới dữ liệu phí  
 thì operater+ sẽ yêu cầu công 2 phép tính lại   
thì số sẽ cộng với nhau còn căng sẽ công vs nhau   
  
a là số 3 + num.a thì con trỏ sẽ trỏ tới 4  
b là số 4 trong căng và num.b sẽ trỏ tới 5 căn bên kia và sẽ trả lai kết quả trên  
  
Operator- trừ giống Operator+ cộng :   
  
Đổi dấu   
Cang operator-()   
  
âm của chính nó chỉ cần bỏ hoặc là đổi từ dương sang âm   
  
Operator\* Nhân  
vd:  
  
thì sau khi khai chuyển   
thì minh code ra được  
  
  
sau đó hàm main:  
  
gọi 1 cái biến c để chứa căng đầu tiên, biến e để chứa số căng thứ 2, sau đó xuất e và c xem thử 2 số đó là 2 số gì,   
gọi 1 căng kết quả thì mặc đinh 2 số a,b = 0  
sau đó cho kết quả bằng kq = c + e

Xuất ra phép cộng,trừ,nhân, đổi dấu  
  
  
Sau đó quay lại vấn đề hàm xuất ở đoạn kết quả   
có vừa cộng vừa trừ, có trường hợp b = 0, hoặc a = 0; hoặc sẽ ra -2   
nên viết thêm hàm xuất  
chỉ thêm một số điều kiện if else cơ bản,  
- dòng đầu:   
 nếu cả a và b = 0 thì nó sẽ xuống dòng   
- dòng 2 :   
nếu a = 0 và b khác 0, thì nó bỏ thằng a ra và chỉ in thằng b căng 7.  
- dòng 3:   
nếu a khác 0 và b = 0 thì nó in ra a.  
- dòng 4 :   
nếu b < 0 thì no sẽ không in ra cái dấu + đảm báo tính thẩm mĩ

Câu 2: Hàm hủy ảo được dùng để đảm bảo rằng hàm hủy của lớp dẫn xuất sẽ được gọi đúng khi mà một đối tượng của lớp dẫn xuất được hủy thông qua con trỏ của lớp cơ sở. Nếu không dùng hàm hủy ảo (ví dụ trong code) thì khi gọi delete ptr; thì chỉ có hàm hủy của lớp DongVat được thực hiện, trong khi hàm hủy của lớp Meo không được gọi. Điều này làm cho bộ nhớ mà ta cấp phát cho Meo không được giải phóng, dẫn đến memory leak.  
Đảm bảo cái hàm hủy của lớp dẫn xuất là lớp kế thước nó được gọi như “DongVat” và cái “ptr” được gọi để delete nó sẽ đảm bảo cả   
cả cái hàm  được gọi theo luôn   
khi gọi delete ptr thì từ Dongvat\*ptr sẽ gọi 1 cái Meo() mới  
khi mà không có hàm hủy ảo khi mình delete ptr này thì nó chỉ gọi duy nhất hàm này:  
  
và sẽ không gọi hàm hủy  
   
điều này làm cho bộ nhớ mình cấp phép cho class Meo không được giải pháo

Câu 3:

1. Không thể tạo ra đối tượng từ lớp absList, vì đây là một lớp trừu tượng. Giải thích về lớp trừu tượng có trong câu 2b đề 2012.
2. Dòng lệnh delete lnkLst làm cho hàm hủy chạy 9 lần. Vì ban đầu, ta tạo ra một đối tượng lnkLst mới, là 1; Sau đó trong vòng lập [1;8] ta chạy hàm addFirst 8 lần, trong mỗi hàm addFirst, điố tượng mới được tạo thêm 1 lần và gán vào subLst, nên là 9 lần.   
   Ban đầu mình đề cập vào hàm hủy trước   
     
   sẽ check xem subLst có phải là 1 con trỏ NULL hay ko nếu không phải nó sẽ delete subLst   
     
   Trong đề câu cuối nó có ghi là delete lnkLst; lnkLst được tạo một lần với giá trị mới là 37   
   sau đó sẽ có một cái vòng lập  
     
   đi từ 1 đến 8 , i<= 8 thì sẽ chạy 8 lần, trong mỗi vòng lập nó sẽ chạy một hàm addFirst của lnkLst và thêm giá trị này vô (i \* i - 7 \* i)  
   thì ở trong hàm addFirst  
     
   tạo một cái linearList mới và nó bỏ giá trị (pId) vô   
   tương tự như thằng này   
   nó sẽ tạo tổng cộng là 9 và nó gọi là một cái danh sách 9 cái giá trị có thể khác nhau hoặc giống nhau  
   khi nó gọi cái biến  này nó sẽ xóa 9 giá trị 9 lần
3. File 2013C3.cpp  
   phương thức countAll của absList phương thức virtual phương thức ảo  
     
   cập nhật phương thức countAll để đếm số phần tử trong danh sách liên kết này   
     
   viết lại hàm countAll bằng phương pháp đệ quy:  
     
   đầu tiên cho count = 1  
   nó sẽ check subLst có bằng NULL hay không   
   nếu bằng NULL nó sẽ tăng một lượng bằng đệ quy của hàm countAll đó   
     
     
     
     
   Hàm showAll để chưa viết mình viết thêm vô   
   copy ở trên   
   
4. Cấu trúc dữ liệu của linearList thực chất là danh sách liên kết đơn. Nhận xét: class khá đơn giản, đủ dùng cho những thao tác cơ bản, nếu muốn ngon hơn thì code thêm hàm remove để xóa phần tử ra khỏi danh sách.